StrangeCompressor

Programação de Software Básico

2019/1

1 Introdução

O algoritmo de Huffman para compressão de arquivos é usado quando sabemos quantas vezes cada caracter aparece em um arquivo que deve ser comprimido. Neste trabalho vamos implementar uma versão um tanto peculiar do algoritmo de compressão, uma vez que, se tudo der certo, teremos arquivos “compactados” bastante maiores que os arquivos de entrada.

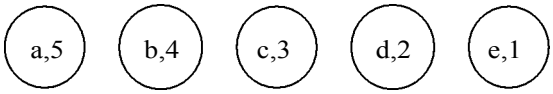
2 Algoritmo de Huffman

O algoritmo é bastante simples:

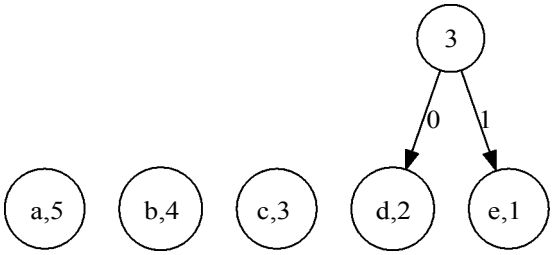
1. O arquivo é analisado e conta se quantas vezes cada caractere (ASCII, de 0 a 255) aparece
2. Cada par (caracter, frequência) é colocado na raiz de uma árvore binária que tem apenas um nodo
3. As duas árvores que tiverem menor frequência são escolhidas e unificadas, transformando se em subárvore esquerda e direita de uma nova árvore binária. A frequência colocada na raiz desta nova árvore é a soma das frequências das suas subárvores
4. O passo anterior é repetido até que exista apenas uma grande árvore binária
5. A codificação é feita associando se bits 0 e 1 para as ligações esquerda e direita de um nodo. O caminho da raiz até um caracter dá a codificação do caracter

Para a string ***aaaaabbbbcccdde*** , a figura abaixo mostra a construção d a árvore e a codificação de cada caracter:

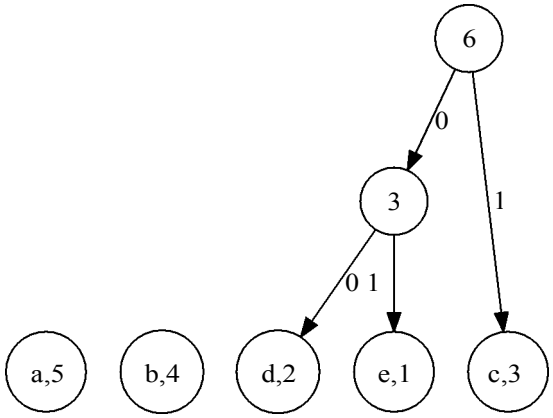
Passo 1:



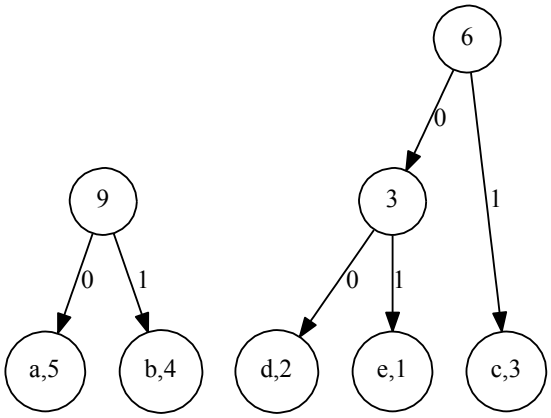
Passo 2



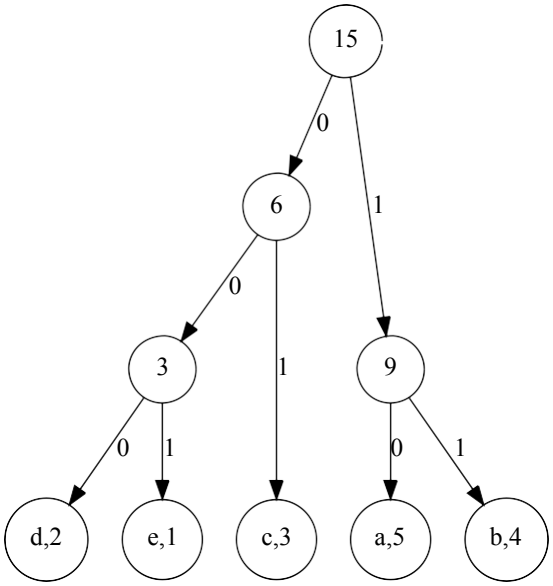
Passo 3:



Passo 4:



Passo 5:



Com isso, obtemos os códigos para cada caracter

a:10 b:11 c:01 d: 000 e: 001

e a string original pode ser representada por *10101010101111111101010100000001*, com apenas 33 bits, ou seja, pouco mais que 4 bytes. A string original pode ser recuperada usando-se estes bits e a tabela de códigos.

---

3 Funcionamento

Sua missão, *should you choose to accept it*, será construir uma aplicação que implemente o algoritmo de Huffman, sendo capaz de codificar/descodificar arquivos textuais (codifique todos os bytes

dos arquivos, de 0 a 255 e não apenas as letras). O programa deve apresentar as seguintes opções de uso:

1. Apenas codificar a árvore, o programa mostra a árvore gerada com os códigos, ordenado do menor para o maior código
2. Gerar um arquivo de saída com a tabela e o arquivo codificado. Atenção, este arquivo deverá ser codificado em decimal (e não em binário). Utilize a extensão ".piz". Estes arquivos devem ter apenas 0s e 1s.
3. Ler um arquivo codificado e apresentar o código original (descompactador)

5 Avaliação

Leia com atenção os critérios de avaliação:

- Pontuação:
  - Compactador:
    - Árvore e tabela de códigos: 4,0 pontos
    - Gravar arquivo codificado (com tabela): 2,0 pontos
  - Descompactador:
    - Descompactar e (re)gerar o arquivio original: 2,0 pontos
  - Organização do código e relatório:
    - código: 1,0 pontos
    - relatório: 1,0 pontos
- Os trabalhos são **em duplas ou individuais**. A pasta do projeto deve ser compactada em um arquivo .zip e este deve ser submetido pelo *Moodle* até a data e hora especificadas.
- Não envie .rar, .7z, .tar.gz - apenas .zip.
- O código deve estar identado corretamente (o *Code::Blocks* **faz isso automaticamente**).
- A nota do trabalho depende da apresentação deste no laboratório, na data marcada. Trabalhos entregues mas não apresentados terão sua nota anulada automaticamente. Durante a apresentação será avaliado o domínio da resolução do problema, podendo inclusive ser possível invalidar o trabalho quando constatada a falta de conhecimento sobre o código implementado.
- **A cópia parcial ou completa do trabalho terá como conseqüência a atribuição de nota ZERO ao trabalho dos alunos envolvidos. A verificação de cópias é feita inclusive entre turmas.**
- **A cópia de código ou algoritmos existentes da Internet também não é permitida**. Se alguma idéia encontrada na rede for utilizada na implementação, sua descrição e referência deve constar no artigo.